AD-A044 903   STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB   F/G 12/1
NUMERICAL ASPECTS OF TRAJECTORY ALGORITHMS FOR NONLINEARLY CONS--ETC(U)
NOV 76 W MURRAY, M H WRIGHT   N00014-75-C-0865

UNCLASSIFIED   SOL-76-29   NL

|OF|

AD
A044903

END
DATE
FILMED
11-77
DDC

NUMERICAL ASPECTS OF TRAJECTORY ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION

BY

WALTER MURRAY and MARGARET H. WRIGHT

TECHNICAL REPORT SOL 76-29

NOVEMBER 1976

Systems Optimization Laboratory

Department of Operations Research

Stanford University

Stanford
California
94305

6

NUMERICAL ASPECTS OF TRAJECTORY ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION.[*]

by

10 Walter Murray[**] [redacted] Margaret H. Wright

9 14 TECHNICAL REP[redacted]T. SOL-76-29

11 Nov[redacted]76 12 14p.

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
Stanford University
Stanford, California

[*]This paper was presented at the Bicentennial Conference on Mathematical Programming, Gaithersburg, Maryland, December 6-8, 1976.

[**]Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, England.

15

Research and reproduction of this report were partially supported by the Office of Naval Research under Contract N00014-75-C-0865; the Energy Research and Development Administration E(04-3)-326 PA #18; and the National Science Foundation Grant DCR75-04544.

408 765

[illegible]

NUMERICAL ASPECTS OF TRAJECTORY ALGORITHMS FOR NONLINEARLY CONSTRAINED OPTIMIZATION

Walter Murray
National Physical Laboratory, Teddington, England

Margaret H. Wright
Stanford University

Abstract

This paper discusses two algorithms for nonlinearly constrained optimization. These algorithms -- the penalty and barrier trajectory algorithms -- are based on an examination of the trajectories of approach to the solution that characterize the quadratic penalty function and the logarithmic barrier function, respectively. Although closely related in principle, the two algorithms display important differences in their implementation as well as in the properties of the generated iterates. The discussion will emphasize the numerical aspects of implementation of the [illegible]jectory algorithms, with particular attention to the choice of reliable methods for carrying out the required computations.

1. Introduction

The problem to be considered in this paper is the following:

$$\text{P1:} \quad \text{minimize} \quad F(x), \qquad x \in E^n$$

$$\text{subject to} \quad c_i(x) \geq 0, \; i = 1,2,\ldots,m,$$

where $F(x)$ and $\{c_i(x)\}$ are prescribed nonlinear functions. The function $F(x)$ is usually termed the "objective function" and the set $\{c_i(x)\}$ is the set of "constraint functions". It will be assumed for simplicity that F and $\{c_i\}$ are twice continuously differentiable, although the methods to be discussed will cope with occasional discontinuities.

The solution of P1 will be denoted by x^*. In all problems to be considered, the first- and second-order Kuhn-Tucker conditions are assumed to be satisfied at x^*, so that there exists a vector λ^* of non-negative Lagrange multipliers, corresponding to the active constraints at x^*, satisfying

$$g(x^*) - \hat{A}(x^*)\lambda^* = 0 \tag{1}$$

where $g(\cdot)$ is the gradient of F, and the columns of $\hat{A}(\cdot)$ are the gradients of the constraints active at x^*.

It is customary to state problem P1 with two kinds of constraints -- [illegible]lity constraints (of the form $c_j(x) = 0$[illegible]l as the inequalities given above in P1. [illegible]stinction will not be made during the present discussion, in order to avoid the introduction of additional notation. The treatment of inequality constraints is typically more complicated than that of equality constraints, and the algorithms to be discussed can deal in an obvious way with equality constraints.

Because the problem P1 can not, in general, be solved explicitly, a popular approach during the past decade has been to transform P1 into a sequence of unconstrained minimization problems. The most common such transformation has been effected by the use of penalty and barrier function methods, which are discussed at length in, for example, Fiacco and McCormick (1968) and Ryan (1974). This paper will not review these methods, but their properties are critical in the derivation of the algorithms to be discussed.

The quadratic penalty function corresponding to the problem P1 is defined by

$$P(x,\rho) = F(x) + \frac{\rho}{2} \sum_{i \in I} (c_i(x))^2 , \tag{2a}$$

where I is a subset of the indices $\{1,2,\ldots,m\}$ (usually, the set of constraints whose values are less than a small positive number), and ρ is a positive scalar termed the "penalty parameter". The quadratic penalty function may also be written as:

$$P(x,r) \equiv F(x) + \frac{1}{2} r^{-1} \sum_{i\in I} (c_i(x))^2 \;, \qquad (2b)$$

where $r = 1/\rho$. Let $x_P^*(r)$ denote an unconstrained minimum of $P(x,r)$.

The logarithmic barrier function corresponding to the problem P1 is given by:

$$B(x,r) \equiv F(x) - r \sum_{i=1}^{m} \ln(c_i(x)) \;, \qquad (3)$$

where r is a positive scalar termed the "barrier parameter". The logarithmic barrier function is defined only at points that strictly satisfy all of the problem constraints. Let $x_B^*(r)$ denote an unconstrained minimum of $B(x,r)$.

Under certain mild conditions, there exists $\hat{r} > 0$ such that for $r < \hat{r}$, $x_P^*(r)$ and $x_B^*(r)$ are continuous functions of r, and:

$$\lim_{r\to 0} x_P^*(r) = x^* \;;$$

$$\lim_{r\to 0} x_B^*(r) = x^* \;.$$

Although penalty and barrier function methods display several good features, they suffer from certain theoretical and numerical defects -- in particular, both require the solution of a theoretically infinite sequence of unconstrained problems. Furthermore, in practice each successive unconstrained sub-problem is more difficult to solve, because the Hessian matrices of the penalty and barrier functions become increasingly ill-conditioned as r approaches zero, and are singular in the limit (see Murray, 1969a).

The continuous lines of minima in E^n defined by $x_P^*(r)$ and $x_B^*(r)$ are termed the "penalty trajectory" and "barrier trajectory" of approach to x^*, respectively. The analysis of these trajectories, and a detailed description of the trajectory algorithms, have been given elsewhere (Murray, 1969a,b; Wright, 1976; Murray and Wright, 1976b); for the purposes of the present discussion, only a brief description of the underlying motivation will be stated.

The trajectory algorithms are based on using the properties of the trajectories to generate a sequence of iterates that lie in a neighborhood of the appropriate trajectory, in order to mimic the approach to x^* of the iterates from a penalty or barrier function method. Because it is possible to characterize a step toward the penalty or barrier trajectory without assuming that the current iterate is in a close neighborhood of x^*, the derivation of the trajectory methods does not display a stringent dependence on properties that hold only in such a neighborhood. Moreover, it is not necessary for any of the iterates to lie exactly on the trajectory (as in a penalty or barrier function method).

At each iteration of a trajectory method, the search direction is computed as a step toward some point on the desired trajectory. The particular point to be aimed for depends on the current value of the penalty or barrier parameter. The solution x^* is also on the trajectories, and the target point will ultimately become arbitrarily close to x^* as the algorithms converge. The penalty or barrier parameter may be adjusted at each iteration of the trajectory methods; however, the choice of the parameter value is not critical, since a step to a neighborhood of a point on the trajectory corresponding to $\bar{r}$ is also in the neighborhood of a point corresponding to $(1+\epsilon)\bar{r}$, where ϵ is small. The numerical procedure for determining the search direction in both algorithms is well-posed, and the approach to the limit of the penalty or barrier parameter does not cause any ill-conditioning.

This paper will emphasize some numerical aspects of implementation of the trajectory methods, with particular attention to the choice of reliable procedures for carrying out the required computations. The emphasis on the details of implementation is deliberate; even within an algorithm that has been designed from the outset to be robust, additional safeguards are necessary to protect against failure or illogical results when the underlying assumptions are not satisfied.

2. Description of Trajectory Algorithms

Only certain key aspects of implementation of the trajectory methods have been selected for discussion in Section 3. Accordingly, the descriptions given here of the penalty and barrier trajectory algorithms are slightly abbreviated, and do not contain all the computational details. A complete description of both algorithms is given in Murray and Wright (1976b).

2.1. Penalty Trajectory Algorithm

2.1.1. Properties of the search direction

For the penalty trajectory algorithm, at each iteration the search direction, p, is (ideally) constructed as the solution of the following quadratic program:

$$\text{QP1:} \qquad \min \ \frac{1}{2} p^T Sp + p^T g$$

$$\text{subject to} \quad \hat{A}^T_p = -\hat{c} - \frac{1}{\rho} \lambda \ ,$$

where $\hat{c}$ denotes the vector of constraints currently considered "active"; $\hat{A}$ is a matrix whose columns are the gradients of the active constraints; λ is an estimate of the Lagrange multiplier vector; ρ is the current value of the penalty parameter; g is the gradient of F; and S is a matrix that approximates the Hessian of the Lagrangian function.

Let Y be a matrix whose columns form an orthogonal basis for the range of the columns of $\hat{A}$, and let Z be a matrix whose columns form an orthogonal basis for the corresponding null space, i.e.,

$$\hat{A}^T Z = 0 \ ,$$

$$Z^T Z = I \ .$$

If $\hat{A}$ has full column rank ($\leq n$), and if the matrix $Z^T SZ$ is positive definite, then the solution of QP1, p^*, can be uniquely expressed as the sum of two orthogonal components:

$$p^* = Yp_R + Zp_N \ .$$

For sufficiently large ρ, the search direction p^* so constructed will always be a descent direction for the quadratic penalty function; the step to be taken along the search direction is then chosen to achieve an acceptable decrease in the penalty function.

2.1.2. Calculation of the search direction

At the beginning of the k-th iteration of the penalty trajectory algorithm, the following vectors and matrices are assumed to be available:

- $x^{(k)}$, an approximation to x^*;
- $c^{(k)}$, the vector of values of $\{c_i(x)\}$ evaluated at $x^{(k)}$;
- $g^{(k)}$, the gradient vector of $F(x)$ evaluated at $x^{(k)}$;
- $A^{(k)}$, the matrix whose columns are the gradients of $\{c_i(x)\}$ evaluated at $x^{(k)}$;
- $S^{(k)}$, an approximation to the Hessian matrix of the Lagrangian function at $x^{(k)}$.

The procedures followed during the k-th iteration to compute the next iterate are:

(1) An "active set" of constraints is determined, containing $\leq n$ elements (see Section 3.1 for a discussion of the case where the active set contains more than n elements). The vector of active constraint values will be denoted by $\hat{c}$, and the matrix whose columns are the gradients of those constraints will be denoted by $\hat{A}$.

(2) Factorize $\hat{A}$ such that

$$Q\hat{A} = \begin{bmatrix} R \\ --- \\ 0 \end{bmatrix} , \qquad Q^T Q = I \ .$$

where R is an upper triangular matrix. Define the matrices Y and Z by partitioning Q as:

$$Q = \begin{bmatrix} Y^T \\ ------ \\ Z^T \end{bmatrix} \ .$$

(3) Determine an estimate, λ, of the Lagrange multiplier vector. If $\hat{A}$ has full solumn rank, λ is the least-squares solution of $\min \| \hat{A}\lambda - g^{(k)} \|_2^2$, and is given by the solution of the triangular

system:

$$R\lambda = Y^T g^{(k)} .$$

If A is rank-deficient, the vector λ will be taken as the minimum-length least-squares solution; it is obtained by extending the factorization of Step (2) to:

$$Q\hat{A}V = \left[\begin{array}{c|c} \bar{R} & 0 \\ \hline \multicolumn{2}{c}{0} \end{array}\right] , \qquad (4)$$

where $\bar{R}$ is a non-singular upper triangular matrix and V is an orthogonal matrix (see Peters and Wilkinson, 1970, for further details).

(4) Determine an appropriate value of the penalty parameter, ρ (Murray and Wright, 1976b).

(5) Compute the vector p_R, as follows. If $\hat{A}$ has full rank, p_R is obtained by solving the linear system:

$$\hat{A}^T p = \hat{A}^T Y p_R = R^T p_R = -\hat{c} - \frac{1}{\rho} \lambda . \qquad (5)$$

In this way, the direction Yp_R *satisfies* the linear equality constraints of QP1. If $\hat{A}$ is rank-deficient, p_R is a least-squares solution of (5), computed using the complete orthogonal factorization (4); the linear constraints of QP1 will then not be exactly satisfied.

(6) Determine the modified Cholesky factorization of the matrix $Z^T S^{(k)} Z$. *With this procedure*, the matrix $Z^T S^{(k)} Z$ is augmented (if *necessary*) by a *positive diagonal matrix*, E, chosen to make $(Z^T S^{(k)} Z + E)$ strictly (numerically) positive definite. Let LDL^T be the computed factorization, so that:

$$LDL^T = Z^T S^{(k)} Z + E ,$$

where E is identically zero if $Z^T S^{(k)} Z$ is sufficiently positive definite (Gill and Murray, 1972a).

(7) Determine the vector $\bar{p}_N$ by solving

$$LDL^T \bar{p}_N = -Z^T g^{(k)} .$$

Test whether:

$$\|\bar{p}_N\| \le M\|p_R\| \quad \text{and} \quad \|p_R\| \le M\|\bar{p}_N\| , \qquad (6)$$

for M a reasonably large positive number (say, 1,000).

(a) If the test (6) is satisfied (as it almost always is in practice), compute p_N by solving:

$$LDL^T p_N = -Z^T(g^{(k)} + S^{(k)} Y p_R) ;$$

then form the search direction as:

$$p = Yp_R + Zp_N .$$

(b) If the test (6) is not *satisfied*, the two orthogonal *portions* of the search direction are not well-scaled, and the following re-scaling procedure is used to adjust for the imbalance.

If $\|\bar{p}_N\| > M\|p_R\|$, define a scaling factor β_1 as

$$\beta_1 = \frac{M\|p_R\|}{\|\bar{p}_N\|} ,$$

and let

$$p = Yp_R + \beta_1 Z\bar{p}_N ;$$

otherwise, define a scaling factor β_2 as

$$\beta_2 = \frac{M\|\bar{p}_N\|}{\|p_R\|} ,$$

and let

$$p = \beta_2 Yp_R + Z\bar{p}_N .$$

(8) Determine a step length, α, that generates an acceptable reduction in the penalty function $P(x, \rho)$, using a safeguarded cubic or parabolic step length algorithm (e.g., the procedure described in Gill and Murray, 1974). Special care must be exercised in the step length algorithm to avoid difficulties if $P(x, \rho)$ is unbounded below along the given search direction (see Section 3.5.1).

(9) Set $x^{(k+1)}$ to $x^{(k)} + \alpha p$, and return to Step (1).

2.2. Barrier Trajectory Algorithm

2.2.1. Properties of the search direction

The search direction of the barrier trajectory algorithm is (ideally) constructed as the solution of the following quadratic program:

$$\text{QP2:} \qquad \min \frac{1}{2} p^T Sp + p^T g$$

$$\text{subject to } \hat{A}^T p = d ,$$

where $d_i = -\hat{c}_i + r/\lambda_i$; $\hat{c}$ denotes the vector of constraints currently considered "active"; $\hat{A}$ is a matrix whose columns are the gradients of the active constraints; λ is an estimate of the Lagrange multipliers corresponding to the active constraints; r is the current value of the barrier parameter; g is the gradient of F; and S is an approximation to the Hessian of the Lagrangian function.

It is essential to achieve a reduction in the barrier function $B(x,r)$ at each iteration, because the barrier function serves as a convenient "merit function" for measuring progress toward x^*. The derivation of the barrier trajectory algorithm indicates that the search direction given by the *solution of QP2 may not always be a descent* direction for $B(x,r)$. Therefore, the null-space component of the search direction may alternatively be computed to minimize a quadratic approximation to the Lagrangian function, independent of the component in the range of the columns of $\hat{A}$. This alternative formulation of the search direction is necessary because of the quite different roles of the penalty and barrier parameters as the solution is approached.

2.2.2. Calculation of the search direction

At the beginning of the k-th iteration of the barrier trajectory algorithm, the same vectors and matrices are available as for the penalty trajectory algorithm. The iterates generated by the barrier trajectory algorithm necessarily lie in the strict interior of the feasible region; this algorithm is intended for use on problems where some or all of the problem functions may be ill-defined or undefined outside the feasible region.

The computational procedures followed during the k-th iteration are:

(1) Determine the set of "active" constraints, denoted by $\hat{c}$ (see Section 3.2); form the matrix $\hat{A}$, whose columns are the columns of $A^{(k)}$ corresponding to the active set. By construction, $\hat{A}$ has $\leq n$ columns.

(2) Factorize $\hat{A}$ such that

$$Q\hat{A} = \begin{bmatrix} R \\ --- \\ 0 \end{bmatrix} ,$$

$$Q^T Q = I ,$$

as before.

(3) Determine the Lagrange multiplier estimate λ by solving:

$$R\lambda = Y^T g^{(k)} ,$$

so that λ is the least-squares solution of $\min \|\hat{A}\lambda - g^{(k)}\|_2^2$; a similar procedure to that given for the penalty trajectory algorithm is followed if $\hat{A}$ is rank-deficient. If one or more components of λ are negative, the constraint corresponding to the most negative is deleted from the *active set*; the modified $\hat{A}$ is then factorized, and the new λ vector calculated for the redefined active set. Since the new $\hat{A}$ is simply the previous $\hat{A}$ with one column deleted, the new factorization can be obtained by a simple updating scheme (Gill, Golub, Murray, and Saunders, 1974).

(4) Determine the barrier parameter, r (Murray and Wright, 1976b).

(5) Determine the vector d according to the following rules. Define $s = \|\hat{c}\| + \|Z^T g^{(k)}\|$, and set $\varepsilon = \gamma\, r/s$, where $\gamma > 1$ (say, 2). Let $\hat{r}$ be the barrier parameter from the previous iteration; then:

$$\text{if } \lambda_i > \varepsilon, \text{ set } d_i = -\hat{c}_i + \frac{r}{\lambda_i} ;$$

$$\text{if } \lambda_i \leq -\varepsilon, \text{ set } d_i = -(1 - \frac{r}{\hat{r}})\, \hat{c}_i ;$$

$$\text{otherwise,} \qquad d_i = -\varepsilon .$$

(6) Compute the vector p_R, which is the solution of the linear system:

$$\hat{A}^T Y p_R = R^T p_R = d .$$

In this way, p_R satisfies the desired linear equality constraints of QP2 for those problem constraints for which λ_i is sufficiently positive; an alternative relationship is satisfied for each "active" constraint that has an insufficiently positive multiplier estimate. Again, the rank-deficient case is treated as for the penalty trajectory algorithm.

(7) Compute the modified Cholesky factorization of $Z^T S^{(k)} Z$ (as in the penalty trajectory algorithm); the factorization will be denoted by LDL^T.

(8) Determine $\bar{p}_N$ by solving:

$$LDL^T \bar{p}_N = -Z^T g^{(k)} .$$

Test whether:

$$\|\bar{p}_N\| \le M\|p_R\| \quad \text{and} \quad \|p_R\| \le M\|\bar{p}_N\| , \qquad (7)$$

for M a reasonably large positive number.

(a) If the test (7) is satisfied, obtain p_N by solving

$$LDL^T p_N = -Z^T(g^{(k)} + S^{(k)} Yp_R) ,$$

and define the trial search direction as:

$$p = Yp_R + Zp_N .$$

If p is not a descent direction for $B(x,r)$, re-define p as:

$$p = Yp_R + Z\bar{p}_N .$$

This latter definition is guaranteed to yield a descent direction for $B(x,r)$.

(b) If the test (7) is not satisfied, then adjust the scaling, as in the penalty trajectory algorithm.

(9) Determine a step length, α, that accomplishes a suitable reduction in $B(x,r)$, using special procedures designed for one-dimensional minimization with respect to the logarithmic barrier function (see Section 3.5.2). During the search procedure, record whether violation of any constraint currently considered "inactive" restricts the step length; if so, this constraint will be added to the active set at the next iteration.

(11) Set $x^{(k+1)}$ to $x^{(k)} + \alpha p$, and return to Step (1).

3. Some Considerations of Numerical Analysis in Implementation of the Trajectory Algorithms

In this section, we consider some selected aspects of implementation of the trajectory methods, from the viewpoints of numerical analysis and algorithm definition. It will be stressed throughout this discussion that an implementation could not achieve practical success if the definition of the algorithm depended critically on properties that hold only in a close neighborhood of x^*; the algorithm should produce sensible results, even when such conditions are not satisfied at the current point.

3.1. Selection of the Active Set for the Penalty Trajectory Algorithm

The "active set" of constraints is defined at each iteration of the penalty trajectory algorithm as the set of constraints whose values are less than a specified small positive number. With this definition, the active set is equivalent to the "violated set", and can easily be determined. Such a strategy is reasonable because the penalty trajectory algorithm is based on properties of the quadratic penalty function, and for a sufficiently large value of ρ, the set of constraints violated at $x^*_P(\rho)$ is identical to the set of constraints active at x^* (Fiacco and McCormick, 1968). After the first few iterations, the active set typically remains fixed for the rest of the computation.

It was noted in the definition of the algorithm that a special procedure is used when more than n constraints are violated at the beginning of an iteration. If more than n constraints are violated, and $\hat{A}$ has full rank, the search direction, p, is chosen to attempt to minimize $\|\hat{c}(x+p)\|^2$, by computing p as the solution of $\min \|\hat{c} + \hat{A}^T p\|^2$. The search direction in this case is calculated as follows:

(1) Factorize $\hat{A}^T$ in the form

$$Q\hat{A}^T = \begin{bmatrix} R \\ --- \\ 0 \end{bmatrix} , \qquad Q^T Q = I ,$$

where R is upper triangular and non-singular.

(2) Solve $Rp = -Y^T\hat{c}$, where the columns of Y are given by the first n rows of Q.

The computational procedure is very similar to the calculation of the Lagrange multiplier estimates, and relies on orthogonal transformations to reduce A^T to upper triangular form.

Normally, the condition that more than n constraints are violated occurs because the current point is a poor estimate of the solution, and does not hold at the next iterate. However, it is conceivable that this condition could exist even at x^*, so that possibly every iteration might be special. In this case, the Hessian matrix of the penalty function is not ill-conditioned as the penalty parameter approaches its limit. This special procedure has the same effect as choosing $r = \infty$ in the usual definition of the algorithm, and hence is equivalent to the Gauss-Newton method.

3.2. Selection of the Active Set for the Barrier Trajectory Algorithm

The criteria for selecting the active set in the barrier trajectory algorithm are not so straightforward as in [illegible] nalty trajectory algorithm. Because the [illegible] trajectory algorithm *is a feasible-poi* [illegible] *all problem constraints are satisfied at* every iteration, and the subset of active constraints must be determined by analysis of the behavior of the constraints as the computation proceeds.

The procedure for determining the initial active set is described in Murray and Wright (1976b), and has been satisfactory on the examples tested. The active set tends to be altered only during the early iterations, because of the possibility of misleading local indications that certain constraints are active. The following rules are used to modify the active set at each iteration:

(1) The constraint corresponding to the most negative Lagrange multiplier estimate (if one exists) is deleted, and the remaining multipliers are modified accordingly.

(2) If any constraint considered active appears to be bounded away from zero as the solution is approached, it is deleted from the active set. This decision is highly dependent on scaling; further discussion is given in Murray and Wright (1976b).

(3) If the step-length algorithm was restricted because a supposedly inactive constraint was violated, this constraint is added to the active set at the beginning of the next iteration, and will not be deleted during that iteration, regardless of the sign of its multiplier estimate.

(4) If the number of active constraints exceeds n following the addition of (3), the constraint with the largest value of $c_i(x)$ is deleted from the active set (a further scaling-dependent decision).

3.3. Use of Orthogonal Factorizations

A factorization involving reduction of $\hat{A}$ to triangular form by application of orthogonal transformations is used in several steps of the trajectory algorithms. Such a factorization is convenient and reliable for computation, and has many advantages over alternative procedures. For example, in some algorithms for solving P1, the matrix $\hat{A}^T\hat{A}$ is formed and used to solve linear systems; the poor numerical properties of this strategy are well-known (see Peters and Wilkinson, 1970), especially the possible squaring of the condition number that may result from the formation of $\hat{A}^T\hat{A}$. Furthermore, if the matrix $\hat{A}$ does not have full rank, $\hat{A}^T\hat{A}$ is singular, and some steps of the algorithm may then be undefined.

Computation of the complete orthogonal factorization of $\hat{A}$ means that steps of the trajectory algorithm can be defined (with relatively little extra work) even if $\hat{A}$ is rank-deficient (see Sections 3.3.1 and 3.3.2). Although only the singular value decomposition can fully reveal the closeness of the columns of $\hat{A}$ to linear dependence, the complete orthogonal factorization provides adequate information in many applications (see Golub, Klema, and Stewart, 1976), since the conditioning of the triangular matrix R serves to indicate the "conditioning" of $\hat{A}$.

3.3.1. Calculation of a Lagrange multiplier estimate

The Lagrange multiplier estimate at each iteration of the trajectory algorithms is computed as a least-squares solution of $\min\|\hat{A}\lambda - g\|_2^2$; this first-order estimate is acceptable, since the local rate of convergence of the trajectory methods is not restricted to the rate of convergence of the multipliers (see Wright, 1976). The orthogonal factorization of $\hat{A}$ can be used to calculate a minimum-length least-squares solution, even when $\hat{A}$ does not have full column rank; this alternative is not possible with techniques that involve forming $\hat{A}^T\hat{A}$.

When reducing $\hat{A}$ to upper triangular form, column interchanges are carried out so that the reduced column of largest magnitude is selected as the next column to be reduced; the matrix is considered to be numerically rank-deficient if the norms of all unreduced columns are less than a prescribed tolerance. In this way, all diagonal elements of R are bounded below by the specified tolerance. Although the ill-conditioning of R does not necessarily reveal itself by the presence of a diagonal element that is very small relative to the largest diagonal element, prevention of a too-small diagonal element is sufficient in many cases to control serious ill-conditioning of R.

3.3.2. Calculation of the search direction

The search direction in both trajectory algorithms is computed in two orthogonal components -- one in the range of the columns of $\hat{A}$, the other in the corresponding null space. This definition results from the characterization that the search direction must satisfy a set of linear equality constraints of the form:

$$\hat{A}^T p = b\ , \qquad (8)$$

where b is some vector depending on the algorithm. If $\hat{A}$ has full rank, these equality constraints uniquely determine the component of p in the range of the columns of $\hat{A}$, which is calculated as follows.

The orthogonal matrix Q that reduces $\hat{A}$ to upper triangular form is explicitly formed, by multiplying out the orthogonal transformations used in the reduction. Once Q is available, its rows, appropriately partitioned, provide the required orthogonal bases for the range and null space of the columns of $\hat{A}$.

In the full rank case, it is straightforward to compute the component of p in the range of $\hat{A}$. Since $p = Yp_R + Zp_N$, the equality constraints (8) imply:

$$\hat{A}^T p = \hat{A}^T(Yp_R + Zp_N) = \hat{A}^T Yp_R = R^T p_R = b\ ,$$

which gives p_R as the solution of a non-singular triangular system.

If $\hat{A}$ is rank-deficient, the component p_R may be obtained as the least-squares solution of $\min\|\hat{A}^T p - b\|^2$, again using the complete orthogonal factorization of $\hat{A}$.

In either case, the calculation of p_R is completely straightforward.

3.4. Approximation of the Hessian of the Lagrangian Function

Alternative techniques for approximating the Hessian of the Lagrangian function under various circumstances will not be discussed in any detail (see Murray and Wright, 1976b, for such a discussion), but we shall consider one key property of the Hessian approximation.

The assumed second-order Kuhn-Tucker conditions imply that the matrix Z^TWZ must be positive definite at x^*, where Z is defined in terms of $\hat{A}(x^*)$, and W is the Hessian of the Lagrangian function; however, W itself need not be positive definite, or even non-singular, at x^* or in any neighborhood of x^*.

Certain approaches to solving P1 impose additional conditions on related matrices -- for example, methods involving augmented Lagrangian functions (see Powell, 1969; Fletcher, 1974) require that the penalty parameter be large enough so that the Hessian of the augmented function is positive definite. In both trajectory algorithms, however, only the projected Hessian, Z^TSZ, must be positive definite at every iteration, in order for the solutions of the quadratic programs QP1 and QP2 to be bounded. The vector p_N is the solution of a linear system:

$$Z^TSZ\ p_N = Z^Td\ , \qquad (9)$$

for some vector d, and should be the step to the minimum of a quadratic function related to the Lagrangian function.

Accordingly, the matrix used to calculate p_N is always maintained as numerically positive definite. When Z^TSZ is updated by a quasi-Newton technique, positive definiteness is maintained by updating the Cholesky factorization of the projected matrix, as in revised quasi-Newton methods for unconstrained minimization (Gill and Murray, 1972b). When Z^TSZ is obtained from exact, or finite-difference approximations to, second derivatives, the modified Cholesky factorization of Z^TSZ is computed in order to solve the system (9). In all cases, the matrix used to solve (9) for p_N is represented as LDL^T, where L is unit lower traingular, and D is a diagonal matrix with all elements strictly positive. Such a procedure assures that the portion of the search direction in the null space of the columns of $\hat{A}$ is always well-defined, and bounded.

3.5. Step-Length Algorithms

3.5.1. Detection and correction of unbounded decrease of penalty function

Even when the problem P1 has a bounded solution, the corresponding penalty function or augmented Lagrangian function may be unbounded below, for arbitrarily large values of the penalty parameter (Powell, 1972). Accordingly, when executing a one-dimensional minimization with respect to a penalty function or augmented Lagrangian function, care must be exercised to avoid the possibility of taking an excessively large step.

In particular, the safeguarded cubic or quadratic step-length algorithms (Gill and Murray, 1974) used in the penalty trajectory algorithm require specification of an upper bound on the step length. In the current implementation of the penalty trajectory algorithm, the upper bound is set to correspond to a step of "reasonable" size, rather than an extremely large value. In some cases, the upper bound may impose an unnecessary limit on the stepsize; however, in general such a restriction will cause no serious loss of efficiency for the overall computation, since the next iteration usually corrects the possible poor scaling of the search direction. This conservative strategy is considered to be justified by the extreme difficulties that result if an enormous step is taken because the penalty function is unbounded below: either the next iterate is completely unreasonable, or a large number of evaluations of the problem functions are required before the unboundedness is detected.

In the penalty trajectory algorithm, it is considered that the penalty function may be unbounded along the given direction if the step taken is the specified upper bound. Almost always, the indicated unboundedness can be eliminated simply by increasing the penalty parameter.

3.5.2. Special techniques for the barrier trajectory algorithm

At each iteration of the barrier trajectory algorithm, a step-length algorithm is executed with respect to the logarithmic barrier function, which thus serves as a "merit function". Several authors (Fletcher and McCann, 1969; Lasdon, et al., 1973) have noted the deficiencies of standard step-length algorithms, which are usually based on approximation by low-order polynomials, when applied to the logarithmic barrier function. Therefore, the step-length algorithm of the barrier trajectory method makes use of special techniques that exploit the known properties of the logarithmic barrier function to allow more efficient estimation of an appropriate step length; these techniques are based on simple approximating functions that contain a logarithmic singularity. Only a small amount of additional calculation is required to fit the special approximating functions, and their use leads to a significant increase in efficiency of the one-dimensional minimization, compared to standard procedures (Murray and Wright, 1976a).

4. Conclusions

The penalty and barrier trajectory algorithms are based on the mathematical properties of the approach to x^* of the successive iterates generated by the quadratic penalty function and logarithmic barrier function, respectively. In theory, these algorithms have several desirable properties -- for example, their derivation does not depend on conditions that hold only in a close neighborhood of x^*, and their rate of convergence in the limit is arbitrarily close to that of linearly constrained Lagrangian algorithms (described in Robinson, 1972; Rosen and Kreuser, 1972). In practice, the current implementations of the trajectory algorithms have been successful on many problems, deliberately including examples for which the ideal assumptions are violated. The results thus far indicate that these algorithms compare favorably with similarly careful implementations of other algorithms to solve P1 (see Wright, 1976, for some typical numerical results).

The overall aim of this paper has been to illustrate some of the considerations of numerical analysis that enter the choice of computational procedures for selected aspects of the trajectory algorithms. Numerical analysis may not play a significant role in the process of verifying that the expected behavior of an algorithm under ideal conditions is displayed numerically. However, it is an elementary fact of numerical analysis that theoretically equivalent mathematical procedures do not yield equivalent, or even close, numerical results, and it is an elementary fact of life that hoped-for conditions are not always satisfied. Considerations of numerical analysis should, therefore, be applied to every aspect of the definition and implementation of optimization algorithms in general.

Acknowledgement

We thank Michael Saunders for his many helpful comments.

Research of this paper was partially supported by the Office of Naval Research under Contract N00014-75-C-0865; the Energy Research and Development Administration E(04-3)-326 PA #18; and the National Science Foundation Grant DCR75-04544. at Stanford University.

References

Fiacco, A.V. and McCormick, G.P. (1968). Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, New York.

Fletcher, R. (1974). "Methods Related to Lagrangian Functions," in Numerical Methods for Constrained Optimization (P.E. Gill and W. Murray, eds.), pp. 219-240, Academic Press, London and New York.

Fletcher, R. and McCann, A.P. (1969). "Acceleration Techniques for Nonlinear Programming," in Optimization (R. Fletcher, ed.), pp. 203-213, Academic Press, London and New York.

Gill, P.E., Golub, G.H., Murray, W., and Saunders, M.A. (1974). Methods for Modifying Matrix Factorizations, Mathematics of Computation, vol. 28, pp. 505-535.

Gill, P.E. and Murray, W. (1972a). Two Methods for the Solution of Linearly Constrained and Unconstrained Optimization Problems, Report NAC-25, National Physical Laboratory.

Gill, P.E. and Murray, W. (1972b). Quasi-Newton Methods for Unconstrained Optimization, Journal Inst. Math. Appl., vol. 9, pp. 91-108.

Gill, P.E. and Murray, W. (1974). Safeguarded Steplength Algorithms for Optimization using Descent Methods, Report NAC-37, National Physical Laboratory.

Golub, G.H., Klema, V., and Stewart, G.W. (1976). Rank Degeneracy and Least Squares Problems, Report CS-76-559, Stanford University.

Lasdon, L.S., Fox, R.L., and Ratner, M.W. (1973). An Efficient One-Dimensional Search Procedure for Barrier Functions, Mathematical Programming, vol. 4, pp. 279-295.

Murray, W. (1969a). Constrained Optimization, Report MA79, National Physical Laboratory.

Murray, W. (1969b). "An Algorithm for Constrained Minimization," in Optimization (R. Fletcher, ed.), pp. 247-258, Academic Press, London and New York.

Murray, W. and Wright, M.H. (1976a). Efficient Linear Search Algorithms for the Logarithmic Barrier Function, Report SOL 76-18, Systems Optimization Laboratory, Stanford University.

Murray, W. and Wright, M.H. (1976b). Trajectory Algorithms for Nonlinearly Constrained Optimization, in preparation.

Peters, G. and Wilkinson, J.H. (1970). The Least-Squares Problem and Pseudo-Inverses, Computer Journal, vol. 13, pp. 309-316.

Powell, M.J.D. (1969). "A Method for Nonlinear Constraints in Minimization Problems," in Optimization (R. Fletcher, ed.), pp. 283-298, Academic Press, London and New York.

Powell, M.J.D. (1972). "Problems Related to Unconstrained Optimization," in Numerical Methods for Unconstrained Optimization (W. Murray, ed.), pp. 29-55, Academic Press, London and New York.

Robinson, S.M. (1972). A Quadratically Convergent Algorithm for General Nonlinear Programming Problems, Mathematical Programming, vol. 3, pp. 145-156.

Rosen, J.B. and Kreuser, J. (1972), "A Gradient Projection Algorithm for Nonlinear Constraints," in Numerical Methods for Non-linear Optimization (F.A. Lootsma, ed.), pp. 297-300, Academic Press, London and New York.

Ryan, D.M. (1974). "Penalty and Barrier Functions," in Numerical Methods for Constrained Optimization (P.E. Gill and W. Murray, eds.), pp. 175-190, Academic Press, London and New York.

Wright, M.H. (1976). Numerical Methods for Nonlinearly Constrained Optimization, Report 193, Stanford Linear Accelerator Center, Stanford, California.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 76-29 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Numerical Aspects of Trajectory Algorithms for Nonlinearly Constrained Optimization		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Walter MURRAY and Margaret H. WRIGHT		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0865 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research, SOL ✓ Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-064
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program Code 434 Office of Naval Research Arlington, VA 22217		12. REPORT DATE November 1976
		13. NUMBER OF PAGES 11
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper discusses two algorithms for nonlinearly constrained optimization. These algorithms -- the penalty and barrier trajectory algorithms -- are based on an examination of the trajectories of approach to the solution that characterize the quadratic penalty function and the logarithmic barrier function, respectively. Although closely related in principle, the two algorithms display important differences in their implementation as well as in the properties of the generated iterates. The discussion will emphasize the numerical aspects of implementation of the trajectory algorithms, with particular attention to the choice of reliable methods for carrying out the required computations.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)